Ma50b Complejidad Computacional **10 de Mayo de 2008**

Pauta Control 2

Prof. Cátedra: M. Kiwi *Prof. Auxiliar: P. Camacho*

PROBLEMA 1:

(i).- Supongamos que L es recursivamente enumerable. Luego, existe M máquina de Turing que reconoce L. Sea M' la máquina de Turing que en la entrada ω simula M y si M entra en un estado de aceptación entonces borra su cinta, escribe 1 en su primera celda y se detiene. Si M en la entrada ω se detiene sin aceptar, entonces M' desplaza su cabeza lectora hacia la derecha indefinidamente. Se verifica fácilmente que la función $f_{M'}$ que calcula M' es parcialmente recursiva y que $f_{M'} : L \subseteq \Sigma_M^* \to \{0,1\}$.

Supongamos ahora que $f : L \subseteq \Sigma^* \to \Sigma^*$ es una función parcialmente recursiva calculada por una máquina de Turing M. Sea M' la máquina de Turing que en la entrada ω simula M y si M se detiene, entonces acepta. Se tiene que si $\omega \in L$, entonces M' acepta ω. Si $\omega \notin L$, entonces M no se detiene en ω y por lo tanto M' no acepta ω. En resumen, M' reconoce L y por lo tanto L es recursivamente enumerable.

(ii.1).- Si $C(M,\omega) \leq (k-1)$, entonces el número de descripciones instantáneas distintas que puede alcanzar M a partir de la entrada ω es a lo más $|Q_M|(k-1)|\Sigma_M|^{k-1}$ (por $|Q_M|$ estados posibles, $k-1$ posiciones factibles de la cabeza lectora y $|\Sigma_M|^{k-1}$ configuraciones de la cinta). Sigue que para decidir si $C(M,\omega) \leq (k-1)$ basta simular M en ω durante $|Q_M|(k-1)|\Sigma_M|^{k-1}$ transiciones. Si durante la simulación M nunca alcanza la celda k de su cinta, tiene que ser porque M se detuvo o está en un loop en el que seguirá por siempre pero que nunca la llevará a alcanzar la celda k de su cinta. Cualquiera sea el caso, se tendrá que $C(M,\omega) \leq k-1$. Lo anterior permite concluir fácilmente que L_k es decidible.

(ii.2).- Bastará establecer que $A_{mT} \leq_m L_{+\infty}$. Para ello, a partir de $\langle M,\omega\rangle$ donde M es máquina de Turing y $\omega \in \Sigma_M^*$ generamos $\langle M',\omega\rangle$ tal que M' es exactamente igual a M salvo que los estados de paro que no son de aceptación han sido sustituidos por una subrutina que hace que M' en vez de detenerse sin aceptar como lo hubiera hecho M, desplaza su cabeza lectora hacia la izquierda indefinidamente. Claramente,

$$\langle M,\omega\rangle \in A_{mT} \iff C(M',\omega) \neq \infty \iff \langle M',\omega\rangle \in L_{+\infty}\,.$$

Dado que $\langle M'\rangle$ se puede generar vía modificaciones locales a partir de $\langle M\rangle$ es fácil ver que implícita en la discusión anterior está una reducción de A_{mT} a $L_{+\infty}$.

PROBLEMA 2:

(i).- Supongamos que existe un lenguaje A recursivamente enumerable y M máquina de Turing con oráculo tal que M^A reconoce (respectivamente decide) L. Sea T una máquina de Turing que reconoce A. Observemos que dado $\sigma \in \Sigma_L^*$ cualquiera, si tenemos acceso al oráculo A_{mT}, entonces podemos decidir pertenencia de σ en A. En efecto, basta consultar al oráculo si $\langle T,\sigma\rangle$ pertenece o no a A_{mT}. En caso afirmativo, sigue que $\sigma \in A$

y en el caso contrario se tiene que $\sigma \notin A$. Luego, la siguiente máquina de Turing con oráculo A_{mT} simula M^A:

$S^{A_{mT}}$ = "En la entrada ω,
Simula M en la entrada ω.
– Si M escribe σ en su cinta de oráculo, entonces escribe $\langle T, \sigma \rangle$ en su cinta de oráculo.
– Acepta, si M acepta.
– Para, si M se detiene."

Es fácil ver que $S^{A_{mT}}$ reconoce (respectivamente decide) L si y sólo si M^A reconoce (respectivamente decide) L.

La implicancia reversa es obvia dado que A_{mT} es recursivamente enumerable.

(ii).- Sea $Z = \left\{ \langle M, \omega \rangle : M^{A_{mT}}(\omega) = acepta \right\}$. Se verifica que $Z \in \Sigma_2$ dado que es reconocido por la siguiente máquina de Turing S con oráculo A_{mT}

$S^{A_{mT}}$ = "En la entrada σ,
Verifica que $\sigma = \langle M, \omega \rangle$ con M máquina de Turing y $\omega \in \Sigma_M^*$.
Simula M con oráculo A_{mT} en la entrada ω
Acepta si M acepta, y se detiene si M para.

Asumiremos que Z es decidible por una máquina de Turing H con oráculo A_{mT} y obtendremos una contradicción. Por (i), esto es suficiente para concluir el resultado deseado. Construiremos ahora una nueva máquina de Turing S con oráculo A_{mT} que usa H como subrutina. Esta máquina recibe una entrada de la forma $\langle M \rangle$ con M máquina de Turing con oráculo A_{mT} y determina lo que hace $M^{A_{mT}}$ en la entrada $\langle M \rangle$. Una vez determinado lo que M hace, la máquina S hace lo contrario, i.e. rechaza si $M^{A_{mT}}$ acepta y no acepta si $M^{A_{mT}}$ acepta. Formalmente,

$S^{A_{mT}}$ = "En la entrada σ,
Verifica que $\sigma = \langle M \rangle$ con M máquina de Turing con oráculo.
Simula H con oráculo A_{mT} en la entrada $\langle M, \langle M \rangle \rangle$ (para la simulación S usa su oráculo)
– Acepta si $H^{A_{mT}}$ no acepta, y rechaza si $H^{A_{mT}}$ acepta."

Resumiendo,

$$S^{A_{mT}}(\langle M \rangle) = \begin{cases} acep., & \text{si } M^{A_{mT}} \text{ no acepta } \langle M \rangle, \\ rech., & \text{si } M^{A_{mT}} \text{ acepta } \langle M \rangle. \end{cases}$$

Se obtiene la siguiente contradicción,

$$S^{A_{mT}}(\langle S \rangle) = \begin{cases} acep., & \text{si } S^{A_{mT}} \text{ no acepta } \langle S \rangle, \\ rech., & \text{si } S^{A_{mT}} \text{ acepta } \langle S \rangle. \end{cases}$$

Luego, H no existe y por lo tanto Z no es decidible por una máquina de Turing con oráculo A_{mT}.

(iii).- Sea $R = \{\langle M \rangle : H(M) = 0\}$. Para establecer que $L \in \Sigma_2$ bastará mostrar que R es recursivamente enu-

merable. En efecto, la siguiente máquina de Turing S con oráculo R decide L,

S^R = "En la entrada ω,

Verifica que $\omega = \langle M_1, M_2 \rangle$ con M_1 y M_2 máquinas de Turing con oráculo.

Para $i \in \{1,2\}$

– Consulta al oráculo acerca de la pertenencia de $\langle M_i \rangle$ en R.

Acepta si exactamente una de las consultas al oráculo se responde afirmativamente."

Para ver que R es recursivamente enumerable basta considerar una máquina de Turing T que en la entrada $\langle M \rangle$ simula, para $n \in \mathbb{N}$, n transiciones de M en cada una de las primeras n (en orden lexicográfico) secuencias $\omega_1, \ldots, \omega_n \in \{0,1\}^*$. La máquina T acepta si para alguna de las simulaciones M se detiene. Es fácil ver que T reconoce R. En efecto, si $\langle M \rangle \in R$, entonces existe un ω y un $n \in \mathbb{N}$ tal que M se detiene en n transiciones en la entrada ω, luego T eventualmente acepta $\langle M \rangle$. Si por el contrario $\langle M \rangle \notin R$, entonces M no se detiene en ninguna entrada, en cuyo caso T tampoco se detiene en la entrada $\langle M \rangle$.

PROBLEMA 3:

(i).- Basta tomar $\langle M \rangle x$ como descripción de x con M máquina de Turing que se detiene tan pronto como se inicializa. Sigue que $K(x) \leq |x| + |\langle M \rangle|$ por lo que tomando $c = |\langle M \rangle|$ se tiene el resultado deseado.

(ii).- Sea $\langle N, \omega \rangle$ una descripción minimal de x. Sea además M una máquina de Turing que verifica que su entrada σ tiene la forma $\langle N', \omega' \rangle$ con N' máquina de Turing y $\omega' \in \{0,1\}^*$, simula N' en ω' hasta que la máquina se detiene con s en su cinta, copia s y se detiene con ss en su cinta. Sigue que $\langle M, \sigma \rangle$ donde $\sigma = \langle N, \omega \rangle$ es una descripción de xx. Luego, si $c = |\langle M \rangle|$ se tiene que

$$K(xx) \leq |\langle M \rangle| + |\langle N, \omega \rangle| = c + K(x).$$

(iii).- Sea $\langle M_y, \omega_y \rangle$ una descripción minimal de y. Sea $\omega = \ll \langle n \rangle_2, \langle M_y, \omega_y \rangle \gg$ y M una máquina de Turing que en la entrada ω la descompone en el prefijo que precede a la primera ocurrencia de 01 y $\langle M_y, \omega_y \rangle$, calcula n a partir de la codificación en binario de n dada por $\langle n \rangle_2$ e y a partir de $\langle M_y, \omega_y \rangle$, y finalmente imprime n copias de y. Sigue que $\langle M, \omega \rangle$ es una descripción de x tal que

$$|\langle M, \omega \rangle| \leq |\langle M \rangle| + 2|\langle n \rangle_2| + K(y).$$

Observando que $|\langle m \rangle_2| \leq \log_2 m + 1$ si $m > 0$ y tomando $c = |\langle M \rangle| + 2$ se obtiene el resultado deseado.

(iv).- Sean $\langle M_x, \omega_x \rangle$ y $\langle M_y, \omega_y \rangle$ las descripciones minimales de x e y. Sea $\omega = \ll \langle K(x) \rangle, \langle M_x, \omega_x \rangle \langle M_y, \omega_y \rangle \gg$ y M una máquina de Turing que en la entrada ω la descompone en el prefijo que precede a la primera ocurrencia de 01 y $\omega' = \langle M_x, \omega_x \rangle \langle M_y, \omega_y \rangle$, calcula $K(x)$ a partir de la codificación en binario de $K(x)$ dada por $\langle K(x) \rangle_2$, usa dicho valor para descomponer ω' en $\langle M_x, \omega_x \rangle$ y $\langle M_y, \omega_y \rangle$, y finalmente determina e imprime x e y a partir de sus descripciones minimales. Sigue que $\langle M, \omega \rangle$ es una descripción de xy tal que

$$|\langle M, \omega \rangle| \leq |\langle M \rangle| + 2|\langle K(x) \rangle_2| + K(x) + K(y).$$

Observando que $|\langle m \rangle_2| \leq \log_2 m + 1$ si $m > 0$ y tomando $c = |\langle M \rangle| + 2$ se obtiene el resultado deseado.

(v).- Basta observar que el número de secuencias binarias distintas de largo n es exactamente igual a 2^n y que el número de descripciones minimales de largo a lo más $n-1$ es $\sum_{i=0}^{n-1} 2^i = 2^n - 1$. Luego, debe existir alguna secuencia binaria de largo n cuya descripción minimal tiene un largo al menos n.